Enhancing Data Analysis with Noise Removal

Hui Xiong, *Member, IEEE*, Gaurav Pandey, Michael Steinbach, *Member, IEEE*, and Vipin Kumar, *Fellow, IEEE*

Abstract

Removing objects that are noise is an important goal of data cleaning as noise hinders most types of data analysis. Most existing data cleaning methods focus on removing noise that is the result of low-level data errors that result from an imperfect data collection process, but data objects that are irrelevant or only weakly relevant can also significantly hinder data analysis. Thus, if the goal is to enhance the data analysis as much as possible, these objects should also be considered as noise, at least with respect to the underlying analysis. Consequently, there is a need for data cleaning techniques that remove both types of noise. Because data sets can contain large amounts of noise, these techniques also need to be able to discard a potentially large fraction of the data. This paper explores four techniques intended for noise removal to enhance data analysis in the presence of high noise levels. Three of these methods are based on traditional outlier detection techniques: distance-based, clustering-based, and an approach based on the Local Outlier Factor (LOF) of an object. The other technique, which is a new method that we are proposing, is a hyperclique-based data cleaner (HCleaner). These techniques are evaluated in terms of their impact on the subsequent data analysis, specifically, clustering and association analysis. Our experimental results show that all of these methods can provide better clustering performance and higher quality association patterns as the amount of noise being removed increases, although HCleaner generally leads to better clustering performance and higher quality associations than the other three methods for binary data.

Index Terms

Data Cleaning, Very Noisy Data, Hyperclique Pattern Discovery, Local Outlier Factor (LOF), Noise Removal

Hui Xiong is with the Management Science and Information Systems Department, Rutgers University, 180 University Avenue, Newark, NJ 07102. E-mail hui@rbs.rutgers.edu. Gaurav Pandey, Michael Steinbach, and Vipin Kumar are with the Department of Computer Science and Engineering, University of Minnesota, 200 Union Steet SE, Minneapolis, MN 55455. E-mail: {gaurav, steinbac, kumar}@cs.umn.edu.

Corresponding Author: Hui Xiong. Phone: 973-353-5261. Fax: 973-353-5003. Email: hui@rbs.rutgers.edu.

I. Introduction

Noise is “irrelevant or meaningless data” [6]. For most existing data cleaning methods, the focus is on the detection and removal of noise (low-level data errors) that is the result of an imperfect data collection process. The need to address this type of noise is clear as it is detrimental to almost any kind of data analysis. However, ordinary data objects that are irrelevant or only weakly relevant to a particular data analysis can also significantly hinder the data analysis, and thus these objects should also be considered as noise, at least in the context of a specific analysis. For instance, in document data sets that consist of news stories, there are many stories that are only weakly related to the other news stories. If the goal is to use clustering to find the strong topics in a set of documents, then the analysis will suffer unless irrelevant and weakly relevant documents can be eliminated. Consequently, there is a need for data cleaning techniques that remove both types of noise.

In some cases the amount of noise in a data set is relatively small. For example, it has been claimed that field error rates for business are typically around 5% or less if an organization specifically takes measures to avoid data errors [28]. However, in other cases, the amount of noise can be large. For example, a significant number of false-positive protein interactions are present in current experimental data for protein complexes. Gavin *et al.* [10] estimate that more than 30% of the protein interactions they detect may be spurious, as inferred from duplicate analyses of 13 purified protein complexes. Although this is an example of a data set that has a large amount of noise due to data collection errors, the amount of noise due to irrelevant data objects can also be large. Examples include the document data sets mentioned earlier [8] and Web data [41], [42]. Therefore, data cleaning techniques for the enhancement of data analysis also need to be able to discard a potentially large fraction of the data.

This paper explores four techniques intended for data cleaning to enhance data analysis in the presence of high noise levels. Three of the techniques are based on traditional outlier detection techniques: distance-based, clustering-based, and an approach based on the Local Outlier Factor (LOF) of an object. More generally, we could exploit any outlier detection approach that assigns each point a value that measures

the degree to which it is an outlier.

The other technique, which is a new method that we are proposing, HCleaner, is a hyperclique-based data cleaner. HCleaner is based on the concept of hyperclique patterns [40], which consist of objects that are strongly similar to each other. In particular, every pair of objects within a hyperclique pattern is guaranteed to have a cosine similarity above a certain level. The cosine similarity measure is also known as the uncentered Pearson's correlation coefficient[1], a measure of association that describes the strength or magnitude of a relationship between two objects. HCleaner filters out all objects that do not appear in any hyperclique pattern.

The framework used for measuring data cleaning performance is based on the impact of the data cleaning on the subsequent data analysis. This approach to evaluation is similar in spirit to that taken by the wrapper technique for subset feature selection [22], which evaluates the choice of a subset of features by its impact on classification performance. Specifically, our hypothesis is that better noise removal results in better data analysis. In this paper, we evaluate data cleaning in terms of its effect on two unsupervised data mining analysis techniques: clustering and association analysis.

Our experimental results show that using HCleaner generally leads to better performance as compared to the outlier based data cleaning alternatives. These other techniques sometimes performed as well or slightly better, but their performance was not as consistent. For instance, the clustering based technique had good performance only when the number of clusters specified matched the actual number of classes in the data. However, this limitation significantly restricts the usefulness of this method.

A. Contributions

The main contributions of this paper are summarized as follows:

- We explore four data cleaning techniques intended to enhance data analysis in the presence of high noise levels. The strengths and weakness of each technique are discussed.

[1]When computing Pearson's correlation coefficient, the data mean is not subtracted.

- One of the four data cleaning techniques is a new data cleaning method, HCleaner, which uses hypercliques to filter out noisy data objects. Our experimental results on several real-world data sets indicate that HCleaner tends to provide better clustering performance and high quality associations than other data cleaning alternatives for binary data.
- We describe a framework for validating data cleaning techniques that is based on the hypothesis that better noise removal schemes lead to better data analysis. Our evaluation of data cleaning techniques is based on their impact on clustering and association analysis.

B. Overview

The remainder of this paper is organized as follows. Section II briefly reviews existing data cleaning techniques. In Section III, we discuss how to exploit existing outlier detection algorithms for noise removal, while in Section IV we present our hyperclique-based data cleaning method. Section V describes our experimental validation approach and Section VI presents the experimental results. Section VII gives our conclusions and indicates possibilities for future work.

II. Data Cleaning Techniques

Data cleaning addresses a variety of data quality problems, including noise and outliers, inconsistent data, duplicate data, and missing values. In this section we briefly review existing work in data cleaning. We separate our discussion into techniques that address the data collection stage and those that focus on the data analysis stage.

A. Data Cleaning Techniques at the Data Collection Stage

At the data collection stage, data cleaning techniques [12], [13], [16], [24] are primarily used to detect and remove errors and inconsistencies from data. Most typical data errors are due to the misuse of abbreviations, data entry mistakes, duplicate records, missing values, spelling errors, outdated codes, etc. [26]. Within this context, one key research topic is the de-duplication problem [17], which is the detection

and removal of duplicate records from a database. The research challenge is that databases contain both exact and inexact duplicates. The inexact duplicates [27] are records that refer to the same real-world entity, but do not have the same values for all fields. One general approach for de-duplicating records follows a filter-and-refine paradigm. In the filtering step, a family of Sorted Neighborhood Methods (SNM) [16] has been proposed to determine which records need to be compared. Then, in the refinement step, actual comparisons are performed to decide whether these records are duplicates or not. Along this line, there is a comprehensive data cleaning system, AJAX [12], which includes four types of data transformations—mapping, matching, clustering, and merging—that can be helpful for eliminating errors, inconsistencies, or duplicates. In addition, AJAX provides a declarative language [13] to specify the flow of logical transformations.

In summary, data cleaning techniques developed at the data collection stage are focused on detecting and removing low-level errors and inconsistencies due to an imperfect data collection process. Indeed, most traditional data cleaning techniques belong to the data collection stage.

B. Data Cleaning Techniques at the Data Analysis Stage

At the data analysis stage, the main purpose of data cleaning techniques is to remove data objects for the purpose of improving the results of the data analysis. Detecting and removing errors is not the key focus. Indeed, the objects being removed may be errors or they may be objects that are irrelevant or only weakly-relevant to the underlying data analysis. In either case, the goal is to remove objects that hinder the data analysis.

An example of data cleaning for error detection and correction is research, within the machine learning community, to identify and eliminate mislabeled training samples for better classification. For instance, Brodley et al. [5] uses consensus filters and majority vote filters to identify and eliminate mislabeled training samples. Their results show that if that the training data set is sufficiently large, then classification accuracy can be improved as more and more suspiciously labeled objects are removed.

Cluster analysis provides an example of data cleaning for the elimination of weakly relevant or irrelevant objects. It is well known that presence of outliers can distort various kinds of clustering, e.g., K-means or hierarchical clustering [19]. As a result, some clustering algorithms [8], [14], [34], [43] attempt to identify which objects are potential outliers during the clustering process. These objects are eliminated and play no role in the final clustering results.

As a final example, Yi et al. [42] uses web Site Style Tree (SST) to capture the common contents and presentation styles of a web site. This approach uses an information based measure to determine which parts of the SST represent noises. The SST is employed to detect and eliminate noise in web pages so that better results for web page clustering and classification can be achieved.

In this paper, we explore data cleaning techniques with a particular focus on noise removal at the data analysis stage. Specifically, we exploit noise removal techniques that are based on outlier detection in order to enhance data analysis in the presence of high noise levels. While our noise removal techniques are based on outlier detection, these techniques are different from outlier detection techniques in two significant ways. First, the notion of an anomaly or outlier implies rareness with respect to the majority of normal objects. However, as this paper demonstrates, eliminating a substantial fraction of all data objects can enhance the data analysis. Second, outlier detection techniques seek to avoid classifying normal objects as outliers. As is also demonstrated by this paper, the elimination of irrelevant or weakly relevant (normal) objects is often essential for enhancing the data analysis.

Because what we are proposing is not simply outlier detection, it would be confusing to refer to the objects that we eliminate as outliers. Indeed, some of them may be normal objects. Instead, we will refer to the objects that are eliminated as noise since this use of the word falls within the general meaning of noise as meaningless or irrelevant data [6].

III. Noise Removal Based on Outlier Detection

In this section, we discuss potential directions for exploiting existing outlier detection techniques for handling data with extremely high levels of noise. Our objective is to improve the results of data analysis

by removing objects that may distort the analysis. Traditionally, outlier detection techniques remove only a small fraction of the objects since, by definition, the number of outliers in the data is small. However, if the amount of noise in the data is large from either a data collection or data analysis viewpoint, then there is a need for data cleaning techniques that remove large amounts of noise. Thus, we consider only outlier detection techniques that assign each object an outlier score that characterizes the degree to which it is an outlier. Such techniques can remove any specified percentage of noise; i.e., we sort the objects according to their 'outlier score' and eliminate the objects with the highest outlier scores until the desired percentage of objects has been eliminated.

In the literature, there are a number of different types of outlier detection methods [18], [37]. In this paper, we employ three methods: distance-based, density-based, and clustering-based. However, any outlier detection technique that assigns a continuous outlier score to each object can be used.

A. Distance based Outlier Detection Methods for Noise Removal

A simple method [2], [21], [3] of detecting outliers is based on the distance measure. An object in a data set D is a distance-based outlier if at least a fraction α of the objects in D are at a distance greater than r. This outlier definition is simple and easy to understand, but can lead to problems when a data set has regions of varying density. In particular, this approach is based on a criterion determined by the global parameters r and α, and cannot take into account the fact that some objects are in regions of high density, while other objects are in regions of low density.

Algorithm 1 shows the pseudo code of our distance-based noise removal algorithm. This algorithm works as follows. For each object in the dataset, the number of objects that lie within a distance r of it is recorded. According to the distance criteria, noise consists of those objects that have the least number of neighbors within a specified radius. Hence, all the objects are sorted in ascending order with respect to the number of neighbors they have. The first $\epsilon\%$ are declared to be noise and are removed from the data set. The complexity of this algorithm is $O(n^2)$, because nearest neighbor sets have to be constructed for

each data object. Note that the cosine similarity measure is used instead of a distance measure, but this changes nothing essential.

Data : Transaction set T, Noise fraction ϵ, Radius r
Result: Set of noise objects N, Set of non-noise objects P
for $i = 1$ *to* n_{trans} **do**
  $T[i].NumWithinDist \leftarrow 0$;
  for $j = 1$ *to* n_{trans} **do**
    if $((j \neq i)\&\&(CosineSimilarity(T[i], T[j]) \geq r))$ **then**
      $T[i].NumWithinDist++$;
    end
  end
end
$T_{sorted} \leftarrow Sort(T, NumWithinDist, ascending)$;
$n_{noise} \leftarrow \epsilon * n_{trans}$;
$N \leftarrow T_{sorted}[1...n_{noise}]$;
$P \leftarrow T_{sorted}[n_{noise} + 1...n_{trans}]$;
return N, P;

Algorithm 1: A distance based noise removal algorithm.

B. Density based Outlier Detection Method for Noise Removal

Another category of outlier detection methods are designed to identify outliers in data sets with varying densities [9], [31], [35]. One of the most influential approaches relies on the *Local Outlier Factor (LOF)* of each object [4]. The LOF of an object is based on the local density of an object's neighborhood, where an object's neighborhood is defined by the $MinPts$ nearest neighbors of the object. $MinPts$ is a parameter that specifies the minimum number of objects (points) in a neighborhood. Objects with a high LOF are treated as outliers. It is the use of the number of neighbors, rather than a specific distance or similarity, that gives the approach its ability to handle data sets with varying densities.

Algorithm 2 shows the pseudo code of our implementation. The key idea is that every object in a data set is an outlier to some extent and this extent is measured using the *Local Outlier Factor* (LOF). The first part of Algorithm 2 computes this factor for each object. This algorithm has a computational complexity of $O(n^2)$, although this can be reduced to $O(nlog(n))$ for low-dimensional data by the use

of efficient multidimensional access methods [11], such as the R* tree. Since the LOF computation has to be iterated over many values of $MinPts$, the associated constant in the complexity may be large.

Because the cosine measure is used instead of a distance measure, the point with the lowest LOF value is the most *unusual* (noisy) point in the data set. Thus, to eliminate the required amount of noise from the data, all the objects are sorted in ascending order with respect to their LOF values, and the first $\epsilon\%$ are declared to be noise. Note that the sorting order here is different from that in the case where distance measures are used to calculate the LOF values.

While the LOF method does not suffer from problems of varying density, there is an issue of how to select parameters, such as $MinPts$. Indeed, since the LOF of each point may vary with the value of the MinPts parameter, it was suggested in [4] that the LOF of each point should be calculated for a range of values of MinPts and one of them chosen according to some criterion. Accordingly, we ran the LOF calculation algorithm for a wide range of values of MinPts, which depended on the size of the data set. For large data sets, this range was wide, e.g., from 10 to 100, while for smaller ones, a smaller range was considered, e.g., from 5 to 25. (The details of this range for all the data sets used in our experiments can be found in Table VI in Section VI.) For each point, the maximum of all the LOF values calculated over this range was chosen as its LOF. This approach, together with the fact noted earlier that the points with the least LOF are the most prominent outliers, implies that a point is labeled as a local outlier only if it is a prominent outlier for many values of MinPts.

C. Clustering based Outlier Detection Methods for Noise Removal

As mentioned earlier, clustering algorithms can detect outliers as a by-product of the clustering process. For instance, Portnoy et al. [30] treat small clusters, which are far away from other major clusters, as sets of outliers. In other words, all objects in such clusters are treated as noise. This method is sensitive to the choice of clustering algorithms and has difficulties in deciding which clusters should be classified as outliers. Another approach, used in [23] and [43], is based on the following hypothesis: once the data

Data : Transaction set T, Noise fraction ϵ, $MinPtsLB$, $MinPtsUB$, $MinPtsStep$
Result: Set of noise points N, Set of non-noise points P
for $n = MinPtsLB;\ n \leq MinPtsUB;\ n+ = MinPtsStep$ **do**
  $MinPts \leftarrow n$;
  for $i = 1$ *to* n_{trans} **do**
    $InterSimilarity[1...n_{trans}] \leftarrow 0$;
    for $j = 1$ *to* n_{trans} **do**
      $InterSimilarity[j] \leftarrow CosineSimilarity(T[i], T[j])$;
    end
    $InterSimilarity[i] \leftarrow 0$;
    $UpdateKDistNeighbors(T[i], InterSimilarity)$ /*UpdateKDistNeighbors finds the k nearest neighbors for transaction T[i] using the similarity vector InterSimilarity*/;
  end
  for $i = 1$ *to* n_{trans} **do**
    $CalculateLRD(T[i])$ /*CalculateLRD calculates the local reachability density (lrd) for transaction T[i] using its k nearest neighbors and their lrd values*/;
  end
  for $i = 1$ *to* n_{trans} **do**
    $latestLOF \leftarrow CalculateLOF(T[i])$ /*latestLOF computes the local outlier factor for T[i] using its lrd value and those of its k nearest neighbors, for the current value of $MinPts$*/;
    $T[i].lof \leftarrow max(latestLOF, T[i].lof)$;
  end
end
$T_{sorted} \leftarrow Sort(T, lof, ascending)$;
$n_{noise} \leftarrow \epsilon * n_{trans}$;
$N \leftarrow T_{sorted}[1...n_{noise}]$;
$P \leftarrow T_{sorted}[n_{noise} + 1...n_{trans}]$;
return N, P;

Algorithm 2: A noise removal algorithm based on the Local Outlier Factor (LOF).

has been clustered, noise objects are the ones that are farthest from their corresponding cluster centroids. In this paper, we will explore a clustering based data cleaner (CCleaner) that is based on this approach.

Algorithm 3 shows the pseudocode of our implementation of the clustering based noise removal method. In our algorithm, data is clustered using a K-means algorithm available in the CLUTO [20] clustering package, and the cosine similarity (distance) of each object from its corresponding cluster centroid is recorded. The top $\epsilon\%$ objects obtained after sorting these objects in ascending (descending) order with respect to this similarity (distance) constitute the noise objects in the data. The overall complexity of the algorithm is the same as that of an execution of K-means and then a linear scan of the data, i.e., $O(kn)$,

where k is the number of clusters and n is the number of points.

An important issue for CCleaner and other clustering based approaches is how to choose the number of clusters. If there is only one cluster, then the cluster based approach becomes very similar to the distance based approach described earlier. On the other hand, if every object is a separate cluster, then the cluster based approach degenerates to the process of randomly selecting objects as outliers. Our experimental results in Section VI show that CCleaner performs well only when the number of clusters is close to the 'actual' number of clusters (classes) in the data set. However, this limitation significantly restricts the usefulness of this method.

Data : Transaction set T, Noise fraction ϵ, Cluster label set C for T
Result: Set of noise points N, Set of non-noise points P
for $i = 1$ *to* $num\_clusters$ **do**
  $cluster\_center[i][1...n_{items}] \leftarrow avg(T[1...n_{trans}], i)$;
end
for $i = 1$ *to* n_{trans} **do**
  $T[i].ClusterCenterSimilarity \leftarrow CosineSimilarity(T[i], cluster\_center[T[i]])$;
end
$T_{sorted} \leftarrow Sort(T, ClusterCenterSimilarity, ascending)$;
$n_{noise} \leftarrow \epsilon * n_{trans}$;
$N \leftarrow T_{sorted}[1...n_{noise}]$;
$P \leftarrow T_{sorted}[n_{noise} + 1...n_{trans}]$;
return N, P;

Algorithm 3: A cluster based noise removal algorithm

IV. HCLEANER: A HYPERCLIQUE-BASED DATA CLEANER

In this section, we propose a hyperclique-based data cleaner (HCleaner). The key idea behind this method is the use of hyperclique patterns [40] as a filter to eliminate data objects that are not tightly connected to other data objects in the data set. A hyperclique pattern is a new type of association pattern that contains objects that are *highly affiliated* with each other; that is, every pair of objects within a pattern is guaranteed to have a cosine similarity (uncentered Pearson's correlation coefficient) above a certain level. If an object is not part of any hyperclique pattern, then it is likely to be relatively unrelated to other objects, and thus, potentially a noise object.

A. Hyperclique Pattern Discovery

We describe the concepts of hyperclique patterns after first introducing the concept on which it is based: the association rule [1].

Association Rules. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items and $T = \{t_1, t_2, \ldots, t_l\}$ be the set of market basket transactions, where each transaction t_i (for $1 \leq i \leq l$) is a set of items and $t_i \subseteq I$. A **pattern** (itemset) is a set of items $X \subseteq I$, and the **support** of X, $supp(X)$, is the fraction of transactions containing X. For example, in Table I, the support of the pattern $\{i_3, i_4\}$ is 3/5 = 60%, since three transactions (t2, t3, t4) contain both i_3 and i_4. A pattern is a **frequent pattern** if the support of this pattern is above a user-specified support threshold. An association rule is of the form $X \rightarrow Y$, and is interpreted to mean that the presence of pattern X implies the presence of pattern Y in the same transaction, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \phi$. The **confidence** of the association rule $X \rightarrow Y$ is written as $conf(X \rightarrow Y)$ and is defined as $conf(X \rightarrow Y) = supp(X \cup Y)/supp(X)$. For instance, for transaction data shown in Table I, the confidence of the association rule $\{i_3\} \rightarrow \{i_4\}$ is $conf(\{i_3\} \rightarrow \{i_4\})$ = supp($\{i_3, i_4\}$)/supp($\{i_3\}$) = 60% / 80% = 75%.

TABLE I
A SAMPLE TRANSACTION DATA SET.

Transactions	Items
t1	i_1, i_2
t2	i_1, i_3, i_4, i_5
t3	i_2, i_3, i_4, i_6
t4	i_1, i_2, i_3, i_4
t5	i_1, i_2, i_3, i_6

Hyperclique Patterns. Unlike frequent patterns, a hyperclique pattern contains items that are strongly correlated with each other. Indeed, the presence of an item in one transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure is specifically designed to capture the strength of this association.

Definition 1: The **h-confidence** of a pattern $X = \{i_1, i_2, \cdots, i_m\}$, denoted as $hconf(X)$, is a measure that reflects the overall affinity among items within the pattern. This measure is defined as $min(conf(\{i_1\} \rightarrow \{i_2, \ldots, i_m\}), conf(\{i_2\} \rightarrow \{i_1, i_3, \ldots, i_m\}), \ldots, conf(\{i_m\} \rightarrow \{i_1, \ldots, i_{m-1}\}))$, where $conf$ is the confidence of association rule as given above.

Example 1: For the sample transaction data set shown in Table I, let us consider a pattern $X = \{i_2, i_3, i_4\}$. We have $supp(\{i_2\}) = 80\%$, $supp(\{i_3\}) = 80\%$, $supp(\{i_4\}) = 60\%$, and $supp(\{i_2, i_3, i_4\}) = 40\%$. Then,

$$\begin{aligned} conf(\{i_2\} \rightarrow \{i_3, i_4\}) &= supp(\{i_2, i_3, i_4\})/supp(\{i_2\}) = 50\% \\ conf(\{i_3\} \rightarrow \{i_2, i_4\}) &= supp(\{i_2, i_3, i_4\})/supp(\{i_3\}) = 50\% \\ conf(\{i_4\} \rightarrow \{i_2, i_3\}) &= supp(\{i_2, i_3, i_4\})/supp(\{i_4\}) = 66.7\% \end{aligned}$$

So, $hconf(X) = min(conf(\{i_2\} \rightarrow \{i_3, i_4\}), conf(\{i_3\} \rightarrow \{i_2, i_4\}), conf(\{i_4\} \rightarrow \{i_2, i_3\}))$ = 50%.

Definition 2: A pattern X is a **hyperclique pattern** if $hconf(X) \geq h_c$, where h_c is a user-specified minimum h-confidence threshold.

TABLE II
EXAMPLES OF HYPERCLIQUE PATTERNS OF WORDS OF THE LA1 DATA SET.

LA1 Dataset		
Hyperclique patterns	Support	H-confidence
{gorbachev, mikhail}	1.4%	93.6%
{photo, graphic, writer}	14.5%	42.1%
{sentence, convict, prison}	1.4%	32.4%
{rebound, score, basketball}	3.8%	40.2%
{season, team, game, play}	7.1%	31.4%

Table II shows some hyperclique patterns identified from words of the LA1 data set at the h-confidence threshold 0.3. The LA1 data set is part of the TREC-5 collection [38] and includes articles from various news categories such as 'financial,' 'foreign,' 'metro,' 'sports,' and 'entertainment.' For instance, in the

table, the hyperclique pattern {season, team, game, play} is from the 'sports' category.

TABLE III
HYPERCLIQUE PATTERNS FROM RETAIL.

Hyperclique patterns	support	h-confidence
{earrings, gold ring, bracelet}	0.019%	45.8%
{nokia battery, nokia adapter, nokia wireless phone}	0.049%	52.8%
{coffee maker, can opener, toaster}	0.014%	61.5%
{baby bumper pad, diaper stacker, baby crib sheet}	0.028%	72.7%
{skirt tub, 3pc bath set, shower curtain}	0.26%	74.4%
{jar cookie, canisters 3pc, box bread, soup tureen, goblets 8pc}	0.012%	77.8%

In addition, Table III shows some of the interesting hyperclique patterns extracted from a real-life `retail` data set. For example, we identified a hyperclique pattern involving closely related items such as Nokia battery, Nokia adapter, and Nokia wireless phone. We also discovered several interesting patterns containing very low support items such as {earrings, gold ring, bracelet}. These items are expensive, rarely bought by customers, and belong to the same product category.

B. Properties of the H-confidence measure

The h-confidence measure has three important properties, namely the anti-monotone property, the cross-support property, and the strong affinity property. Detailed descriptions of these three properties were provided in our earlier paper [40]. The anti-monotone and cross-support properties form the basis of an efficient hyperclique mining algorithm that has much better performance than traditional frequent pattern mining algorithms, particularly at low levels of support. Here, we provide only a brief summary of the strong affinity property, which is key to developing a good data cleaning scheme.

- **The strong affinity property** guarantees that if a hyperclique pattern has an h-confidence value above the minimum h-confidence threshold, h_c, then every pair of items within the hyperclique pattern must have a cosine similarity (uncentered Pearson's correlation coefficient) greater than or equal to h_c. As a result, the overall affinity of hyperclique patterns can be controlled by properly setting an h-

confidence threshold. Note that the definitions of Pearson's correlation coefficient and uncentered Pearson's correlation coefficient are as follows.

– Pearson's Correlation Coefficient:

$$S(x_1,\, x_2) = \frac{\sum_{k=1}^{n}(x_{1k} - \overline{x_1})(x_{2k} - \overline{x_2})}{\sqrt{\sum_{k=1}^{n}(x_{1k} - \overline{x_1})^2 \sum_{k=1}^{n}(x_{2k} - \overline{x_2})^2}}$$

– Uncentered Pearson's Correlation Coefficient:

$$S(x_1,\, x_2) = \frac{\sum_{k=1}^{n} x_{1k}x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2 \sum_{k=1}^{n} x_{2k}^2}}$$

Every association rule derived from one hyperclique pattern will have a confidence value equal to or greater than the h-confidence value of this hyperclique pattern [39]. For instance, assume that the h-confidence for the hyperclique pattern $X = \{A, B, C\}$ is 0.8, then the confidence of any association rule derived from the pattern X should be greater than or equal to 0.8.

C. Why is the hyperclique pattern a good candidate for removing noise objects?

The strong affinity property discussed above indicates that objects that form a hyperclique pattern are highly related to each other. The degree of relationship is dependent upon the h-confidence threshold; the higher the threshold, the stronger the relationship. In fact, in data sets with class labels, hyperclique patterns with sufficiently high h-confidence thresholds tend to include objects from the same class. This is illustrated in Figure 1, which shows, for the LA1 document data set, the average entropy of the discovered hyperclique patterns for different minimum h-confidence and support thresholds. (Characteristics of the LA1 data set are presented in Table IV in Section VI.) Note that when the minimum h-confidence threshold is zero, we actually have frequent patterns instead of hyperclique patterns.

As Figure 1 shows, when the minimum h-confidence threshold increases, the entropy of hyperclique patterns decreases dramatically. For instance, when the h-confidence threshold is higher than 0.25, the

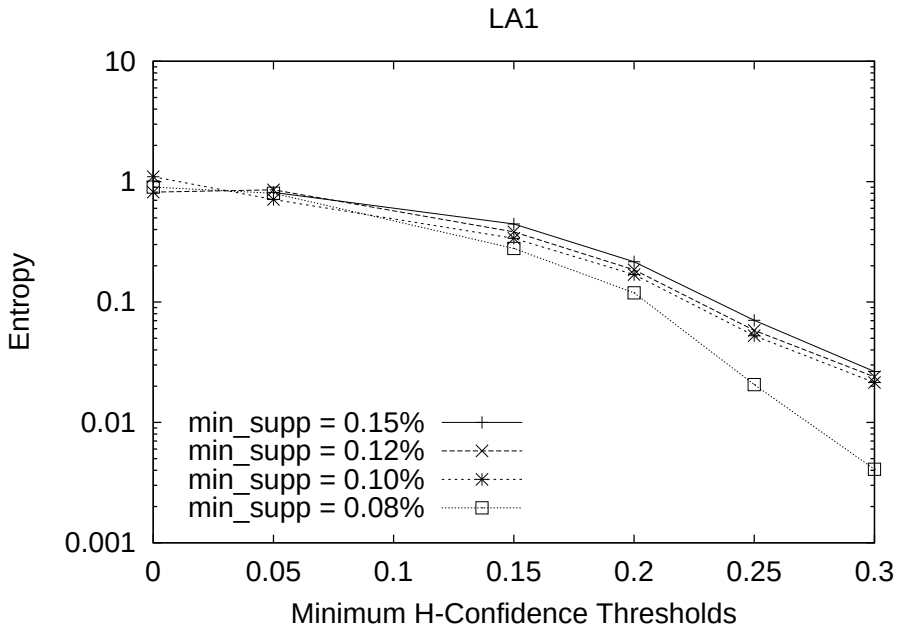


Fig. 1. The cluster nature of hyperclique patterns on the LA1 data set.

entropy of hyperclique patterns will be less than 0.1 at all the given minimum support thresholds. This indicates that, at high enough h-confidence thresholds, hyperclique patterns tend to include objects from the same class. In contrast, the entropy of frequent patterns is high—close to 1—for all the given minimum support thresholds. This means that frequent patterns tend to include objects from different classes.

Another trend that can be observed in Figure 1 is that, with the decrease of the minimum support thresholds, the entropy of hyperclique patterns from the LA1 data set trends downward. This indicates that high affinity patterns can appear at very low levels of support. However, frequent pattern mining algorithms have difficulty in identifying frequent patterns at low levels of support. In contrast, the hyperclique pattern mining algorithm has much better performance at low levels of support [40]. In fact, for many data sets, hyperclique patterns can be found even for support thresholds equal to zero. If we want to discover high-affinity patterns occurring at low levels of support, then the hyperclique pattern is a better choice.

D. Hyperclique based Data Cleaner (HCleaner)

In this subsection, we introduce the hyperclique based data cleaner (HCleaner). The basic idea of HCleaner can be summarized as follows: find all hyperclique patterns (for a given support and h-confidence

threshold) and eliminate any objects that are not a part of any hyperclique pattern. The set of hyperclique patterns for any data set depends upon the value of support and h-confidence thresholds. Wherever possible, we set the support threshold to be zero and employ the h-confidence threshold to control the number of objects that are designated as noise. In some data sets, however, setting the support threshold to zero leads to an explosion in the number of hyperclique patterns. For this reason, we use a low support threshold that is high enough to reduce the number of hyperclique patterns to a manageable level.

Data : Transaction set T
Result: Set of noise points N, Set of non-noise points P, Noise fraction ϵ
$HC \leftarrow HypercliqueMiner(T')$ //HC: the hyperclique set;
$T[1...n_{trans}].covered \leftarrow false$;
$num\_hc \leftarrow size(HC)$;
for $i = 1$ *to* num_hc **do**
  for $j = 1$ *to* n_{trans} **do**
    if $((!T[j].covered)\&\&contains(T[j], HC[i]))$ **then**
      $T[j].covered \leftarrow true$;
    end
  end
end
$N \leftarrow \{\}$;
$P \leftarrow \{\}$;
for $i = 1$ *to* n_{trans} **do**
  if $T[i].covered$ **then**
    $P \leftarrow P \cup T[i]$;
  end
  else
    $N \leftarrow N \cup T[i]$;
  end
end
$\epsilon \leftarrow \frac{|N|}{n_{trans}}$;
return N, P, ϵ;

Algorithm 4: A Hyperclique Based Data Cleaner (HCleaner) Algorithm.

Algorithm 4 shows the pseudocode of our implementation of HCleaner. This algorithm works as follows. We first derive all size-3 hyperclique patterns at a given h-confidence threshold h_c from the transaction set T', where T' is the transpose of the original transaction data T, since we are interested in clustering objects instead of attributes. The noise objects are simply those which are not a member of any of these hyperclique patterns. In other words, for any identified noise object, we cannot find two other objects

which have pairwise cosine similarity with this object above the h-confidence threshold, h_c. Indeed, the h-confidence threshold specifies the fraction of noise data objects. If we fix the support threshold, a higher h-confidence threshold means that more objects will be labeled as noise. Therefore, the noise percentage increases as the h-confidence threshold increases.

In the algorithm, we deliberately selected only size-3 hyperclique patterns. Our rationale for this choice is as follows. For most data sets, there is usually a very large number of size-2 hyperclique patterns. Even a noise object can easily have a strong correlation with another object. Hence, size-2 hyperclique patterns may contain spurious pairwise connections. Instead, we use size-3 hyperclique patterns to help ensure that the connection between objects is not spurious. Indeed, if an object appears in a size-3 hyperclique pattern, it means that there are at least two other objects which have a guaranteed pairwise similarity with this object. In addition, there tend to be very few hyperclique patterns with more than three objects unless the h-confidence threshold is very low, but a low h-confidence threshold may not capture strongly related objects. Our parameter studies in Section VI also indicate that size-3 hyperclique patterns provide a good compromise. However, more work is needed to determine if this is the optimal choice.

Computation Analysis: The major computation cost of HCleaner is from the computation for size-3 hyperclique patterns. Compared to frequent pattern mining algorithms, the hyperclique miner is very efficient in identifying hyperclique patterns and is scalable to very large data sets [40]. In addition, HCleaner does not need to go beyond size-3 hyperclique patterns and, consequently, there is no combinatorial growth of the pattern space. As a result, HCleaner is a very efficient and scalable algorithm.

Finally, the fraction of data to be labeled as noise is not an input to the algorithm, but a result of it. To allow comparison with the other noise removal algorithms, which do allow direct specification of the fraction of noise, we proceed as follows. For different levels of h_c, we determine the fraction of noise points produced by HCleaner. We then use this value as an input to the other noise removal algorithms studied. Hence, a single parameter h_c drives the entire noise removal process.

V. Validation Methods for Data Cleaning Techniques

Just as we can divide data cleaning techniques into two categories based on different stages of the data life cycle—techniques at the data collection stage and techniques at the data analysis stage—we can also divide validation methods into two categories with respect to these two stages. At the data collection stage, where we are focused on detecting and removing errors and inconsistencies from data, one straightforward method for validating the correctness and effectiveness of a data cleaning technique at the data collection stage is to conduct a manual inspection of the data or samples from the data. However, this method is not of much practical use. A promising alternative for data with class labels is to use a supervised learning framework for automatically validating the performance of data cleaning techniques. More specifically, the raw data is divided into training and test data and the effectiveness of data cleaning techniques are then measured by using supervised learning metrics, such as $recall$ and $false\text{-}positive\ errors$ [24].

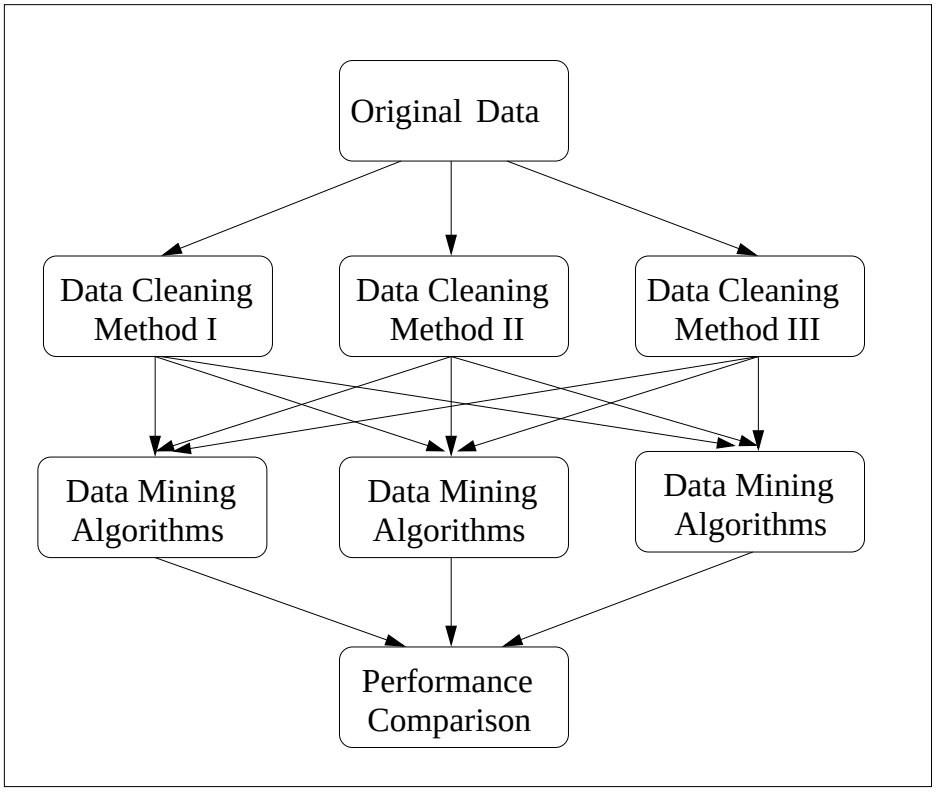


Fig. 2. A Data Mining Framework for Validating Data Cleaning Techniques at the Data Analysis Stage.

There is limited research on validation methodologies for data cleaning techniques at the data analysis stage, especially those that focus on identifying objects that distort the subsequent analysis. Indeed, the

focus of data cleaning research is primarily on the data collection stage. To address this gap, we propose a data mining framework, shown in Figure 2, for validating data cleaning techniques that is based on the hypothesis that better interpretations, models, and decisions can be obtained by better noise removal. Thus, we evaluate the effectiveness of data cleaning by evaluating the results of data mining techniques on cleaned data. For this paper, we only perform this evaluation using unsupervised learning techniques such as clustering analysis and association analysis.

Clustering Evaluation Measures. To evaluate the quality of the clusters produced by the different clustering techniques, we employed two commonly used measures of clustering quality: entropy and the F-measure [23]. Both entropy and the F-measure are 'external' criteria; i.e., they use external information—class labels in this case. Entropy measures the purity of the clusters with respect to the given class labels. Thus, if all clusters consist of objects with only a single class label, the entropy is 0. However, as the class labels of objects in a cluster become more varied, the entropy increases. The F-measure also measures cluster quality, but attains its maximum value when each class is contained in a single cluster, i.e., clusters are pure and contain all the objects of a given class. The F-measure declines as we depart from this 'ideal' situation. Formal definitions of entropy and the F-measure are given below.

Entropy. To compute the entropy of a set of clusters, we first calculate the class distribution of the objects in each cluster; i.e., for each cluster j we compute p_{ij}, the probability that a member of cluster j belongs to class i. Given these probabilities, the entropy of cluster j is calculated using the standard entropy formula

$$E_j = -\sum_i p_{ij} log_2(p_{ij}), \tag{1}$$

where the sum is taken over all classes. The total entropy for a set of clusters is computed as the weighted

sum of the entropies of each cluster, as shown in the equation

$$E = \sum_{j=1}^{m} \frac{n_j * E_j}{n}, \tag{2}$$

where n_j is the size of cluster j, m is the number of clusters, and n is the total number of data points.

F-measure. The F-measure combines the precision and recall concepts from information retrieval [33]. We treat each cluster as if it were the result of a query and each class as if it were the desired set of documents for a query. We then calculate the recall and precision of that cluster for each given class as follows:

$$Recall(i, j) = n_{ij}/n_i \tag{3}$$

$$Precision(i, j) = n_{ij}/n_j \tag{4}$$

where n_{ij} is the number of objects of class i that are in cluster j, n_j is the number of objects in cluster j, and n_i is the number of objects in class i. The F-measure of cluster j and class i is then give by the equation

$$F(i, j) = \frac{2 * Recall(i, j) * Precision(i, j)}{Precision(i, j) + Recall(i, j)}. \tag{5}$$

Association Analysis Evaluation Measures. There are several possible approaches for evaluating the quality of association patterns produced by the association mining algorithms. We could either measure the quality of the association patterns generated, or the quality of the association rules generated from these patterns. In this paper, we employed the IS measure [36] for association patterns. For a pair of items, the IS measure is defined as follows,

$$IS(\{A, B\}) = \sqrt{conf(\{A\} \rightarrow \{B\}) conf(\{B\} \rightarrow \{A\})} = \frac{supp(A, B)}{\sqrt{supp(A) supp(B)}}. \tag{6}$$

$IS(\{A, B\})$ is equivalent to the cosine of the angle between A and B, and hence is an effective measure of the affinity of two items. This measure can be easily extended to the case of an association pattern

$X = \{I_1, I_2, \ldots, I_n\}$ of length n, where $I_1, I_2, \ldots, I_n$ are items, as follows,

$$IS(X) = \frac{supp(I)}{\sqrt{supp(I_1)supp(I_2)\ldots supp(I_n)}} \tag{7}$$

The IS measure has many desirable properties such as a monotonic increase with the support of an item set or the supports of the items constituting the item set, and invariance with the null addition operation. However, since the number of items in an association pattern is unbounded, the IS of an association pattern is also unbounded. Thus, the average IS of a set of item sets is not statistically robust and is not suitable for measuring the quality of a set of association patterns, since the mean could be distorted significantly by a very large IS value. For this reason, we use the median of the IS of a set of association patterns as the quality measure for a set of association patterns.

VI. Experimental Results

Employing the validation methodology described above, we conducted an experimental study to compare HCleaner, CCleaner, and the two previously discussed noise removal techniques that we derived from the LOF and distance based outlier detection algorithms. Specifically, we used these four techniques to remove increasing amount of noise from the data and then applied clustering and association analysis to the cleaned data. These results were evaluated by measuring the entropy and the F-measure of the resulting clusters and the median of the IS of the resulting association patterns.

We begin by describing our experimental setup—the data sets used and our evaluation process. We then present the performance of clustering analysis as increasing numbers of noise objects are removed. We also present a performance study for association analysis. We conclude this section with a sensitivity analysis of the two best approaches, HCleaner and CCleaner. In particular, HCleaner is evaluated for different sizes of hyperclique patterns and CCleaner is evaluated for different numbers of clusters.

A. Experimental Setup

Experimental Data Sets. For our experiments, we used real-world data sets from several different application domains, namely, document data sets, the Internet advertisement data, and microarray gene expression data. The document data sets are the LA1, WAP, RE0, OH8 and WEST5 data sets, which are widely used in document clustering research. The LA1 and OH8 data sets are a part of the TREC-5 collection [38] and contain news articles. The WEST5 data set came from the Thompson Publishing Group and was derived from legal documents. The RE0 data set is from the Reuters-21578 text categorization test collection Distribution 1.0 [25]. The WAP data set is from the WebACE project (WAP) [15], where each document corresponds to a web page listed in the subject hierarchy of Yahoo!. For all document data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [29]. In addition, we used a binary data set called ADS (Internet Advertisements) from the UCI Machine Learning repository[1]. The ADS data set represents a set of possible advertisements on Internet web pages. Some characteristics of these data sets are shown in Table IV.

TABLE IV
CHARACTERISTICS OF DATA SETS.

Data Set	LA1	OH8	RE0	WAP	WEST5	ADS
#Documents	3204	839	1504	1560	311	3279
#Words	31472	2836	11465	8460	1156	1555
#Classes	6	10	13	20	10	2
Source	TREC-5	TREC-5	Reuter	WebAce	Thompson	UCI ML Repository

TABLE V
CHARACTERISTICS OF THE YEAST GENE EXPRESSION DATA SET.

Data Set	Yeast
#Samples	79
#Genes	2467
#Classes	8
Source	Eisen et al (1998)

[1]http://www.ics.uci.edu/∼mlearn/MLRepository.html

We also used gene expression data from the biology domain. The microarray data set includes the Yeast data of Eisen et al. [7]. Some characteristics of this data set are listed in Table V.

Experimental Tasks. The purpose of our experiments was to answer the following questions:

1) What is the impact of noise removal on the clustering performance?
2) What is the impact of noise removal on the results of association analysis?

Experimental Procedures. For the purpose of comparison, clustering and association analysis was performed on both the raw data and the data obtained after removing the specified amount of noise. The approaches used for noise removal were the following: LOF, Distance-based outliers, random selection of noise objects, CCleaner, and HCleaner. The K-means algorithm of the clustering package CLUTO [20] was used for clustering and the number of clusters was set to the actual number of classes for the data. Entropy and the F-measure was used to compare the quality of the clusters obtained and the median of the IS was used to compare the quality of the resulting association patterns

Figure 3 shows the experimental evaluation process for data analysis. First, size-3 hyperclique patterns were extracted from data sets at the specified support and h-confidence thresholds. We found hyperclique patterns among documents for document data sets and among samples for gene expression data sets. All the objects covered by hyperclique patterns were used for subsequent clustering and association analysis.

For the remaining three techniques, the percentage of noise objects was specified as a parameter. This percentage was determined by using the percentage of objects that did not appear in any hyperclique patterns. In this way, we guaranteed that the same number of objects were eliminated by every noise removal technique. The remaining objects were used in the subsequent clustering and association analysis. Table VI shows our parameter settings for data analysis with different noise removal techniques.

An important aspect of the experimental procedure is the use of binarized versions of some of the data sets. The motivation for this is the use of association analysis for various portions of the experimental process. For instance, HCleaner is based on association analysis, and thus, requires a binary data set as input. Likewise, when the effect of noise removal is evaluated for association analysis, binary data sets

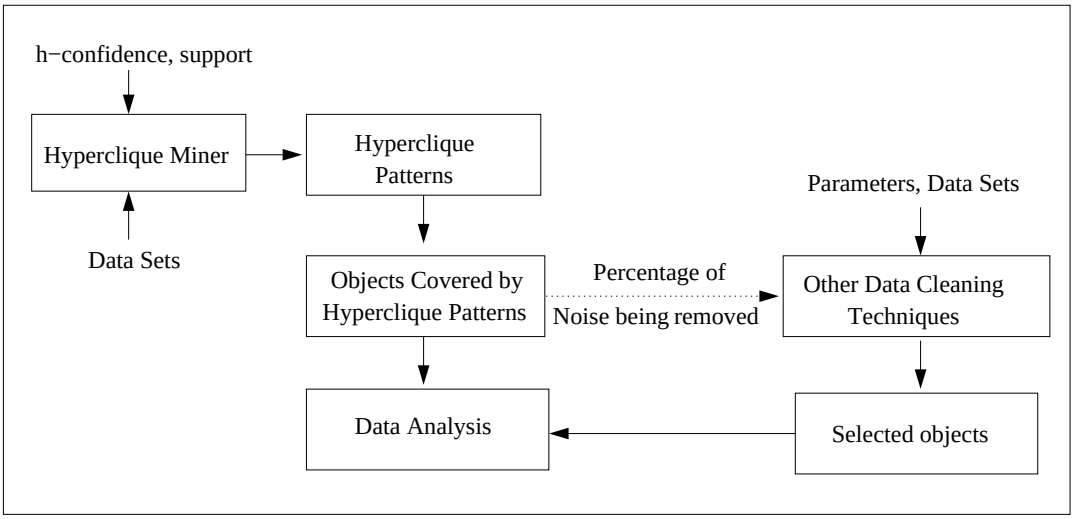


Fig. 3. The Experimental Evaluation Process for Data Analysis.

TABLE VI
THE EXPERIMENTAL PARAMETERS FOR DATA ANALYSIS.

Parameter / Data Set	Range of MinPts for LOF	Similarity threshold for Distance	Number of clusters for clustering	Support for Hypercliques	Range of H−confidence for Hypercliques (%)	Fraction of median support used
WAP	10−100	0.10	20	0.004	5−35	N.A.
RE0	10−100	0.10	13	0.0025	5−35	0.25
OH8	10−100	0.10	10	0.000	8−18	0.125
WEST5	5−25	0.10	10	0.000	25−55	0.50
ADS	10−50	0.10	6	0.000	35−100	0.75
Yeast	5−15	0.50	8	0.000	30−60	N.A.

must be used. However, note that binary versions of the data sets are used only when necessary. Thus, all noise removal techniques except HCleaner use the original data during the noise removal process, and the original data is used for the clustering that is performed to evaluate the impact of noise removal. Through this procedure, we ensure fairness among all the techniques with respect to the input data sets. For completeness, we mention that the document data sets were binarized by converting any non-zero entry into a 1, while the gene expression data was binarized using a standard procedure in which the mean μ was calculated for each gene expression variable, and then any data greater than μ was transformed to a 1 while all other data was changed to a 0.

B. The Impact of Noise Removal on Clustering Performance

In this subsection, we evaluate the impact of noise removal on the results of clustering analysis. For our experiments, we used the document and microarray gene expression data sets described above. Because clustering performance can vary among data sets with different data characteristics, we purposely chose these data sets from different application domains.

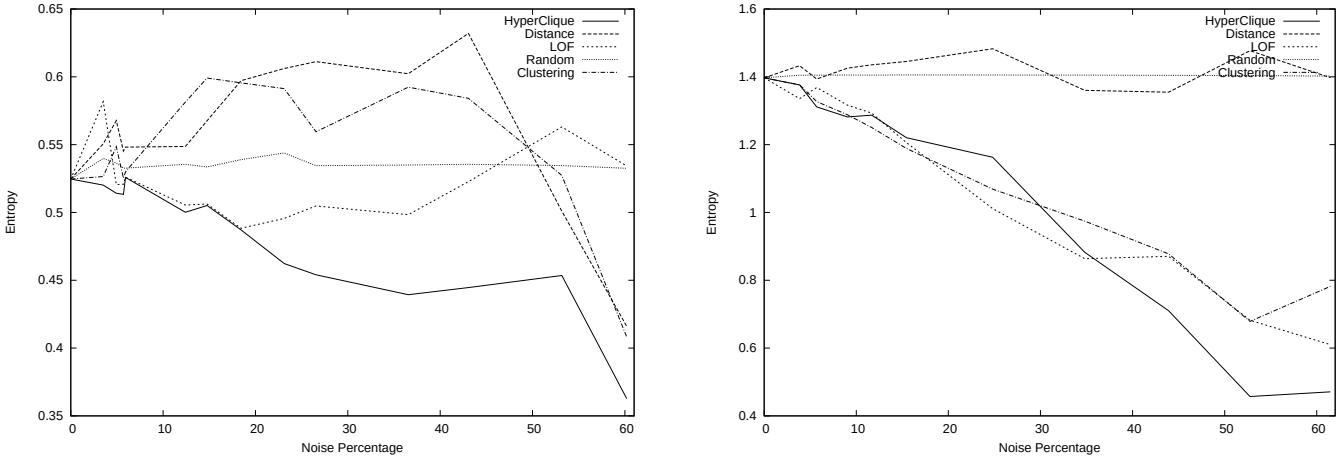
Figures 4 (a) and (b) show the clustering performance on the ADS data set and the WAP data set respectively. Both figures show that if we randomly remove data objects, the entropy is almost unchanged. In other words, clustering performance is not affected by the elimination of random objects. In contrast, as the percentage of noise objects removed by LOF, CCleaner, and HCleaner increases, the entropy generally goes down, i.e., clustering performance improves as more and more noise or weakly-relevant objects are removed. For the ADS data set, we observe that HCleaner provides the best clustering results compared to other noise removal techniques across all experimental cases. For the WAP data set, there are only small performance differences among CCleaner, HCleaner, and LOF when the percentage of noise objects is lower than 30%. However, HCleaner yields significantly better clustering performance as the percentage of objects being removed is increased.

Figures 5 (a) and (b) show clustering performance for the OH8 and WEST5 data sets, respectively. For the OH8 data set, we can observe that CCleaner generally performs better than other noise removal techniques. However, the performance difference between CCleaner and HCleaner is not significant. For the WEST5 data set, there are only small performances difference among CCleaner, HCleaner, and LOF when the percentage of noise objects is lower than 30%. However, HCleaner yields significantly better clustering performance as the percentage of objects being removed is increased. These two data sets also support the observation that the random elimination of objects does not affect clustering performance.

Figure 6, illustrates the impact of noise removal techniques on clustering analysis for the gene expression data sets. In Figure 6, we observe that HCleaner leads to the best clustering performance when the percentage of noise objects is greater than 12%, which covers a typical noise level for gene expression

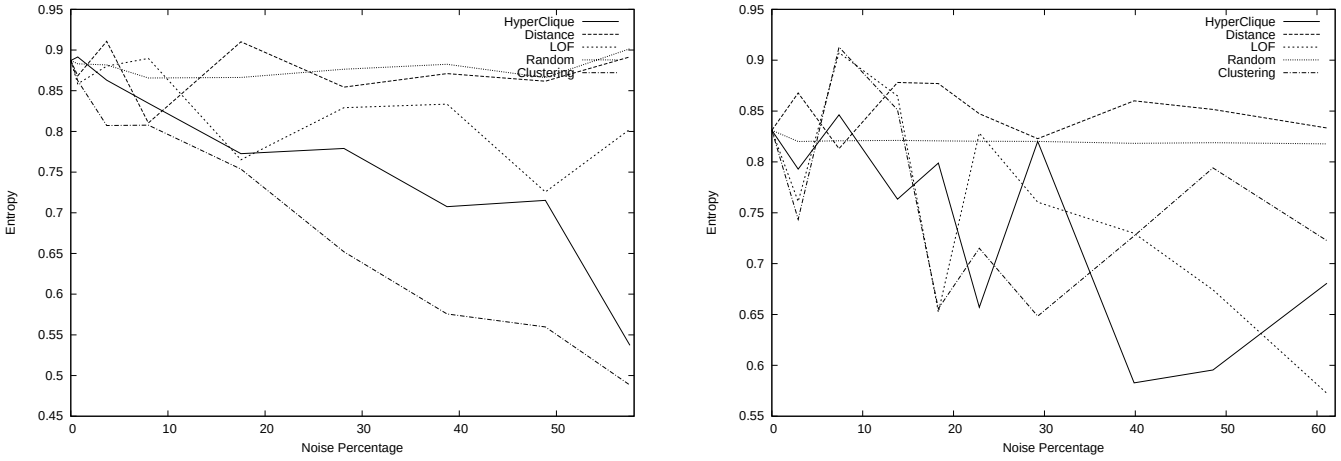
data. Also, we observe that HCleaner has a poor performance when the percentage of noise objects is less than 10%.

Besides entropy, we also applied the F-measure for evaluating the performance of noise removal techniques on the clustering analysis. Figure 7 shows the impact of noise removal techniques on the performance of clustering analysis for the ADS data set. As the figure shows, HCleaner tends to have better (higher) F-measure values than other noise removal techniques for the most experimental cases.



(a) The ADS document data set. (b) The WAP document data set.

Fig. 4. The impact of noise removal techniques on the performance of clustering analysis for ADS and WAP in terms of entropy.



(a) The OH8 data set. (b) The WEST5 data set.

Fig. 5. The impact of noise removal techniques on the performance of clustering analysis for OH8 and WEST5 in terms of entropy.

In summary, regardless of the data set, HCleaner tends to be the best or close to the best technique for improving clustering performance for binary data, while LOF and CCleaner have a competitive

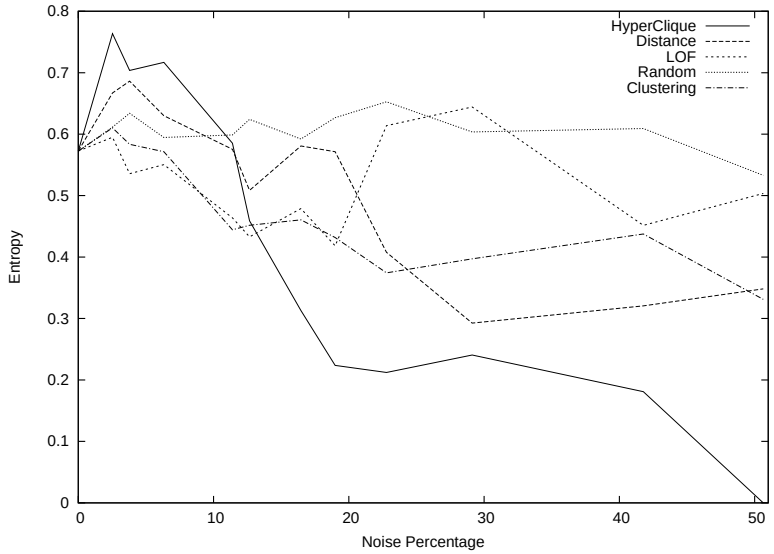


Fig. 6. The impact of noise removal techniques on the performance of clustering analysis for the yeast gene expression data in terms of entropy.

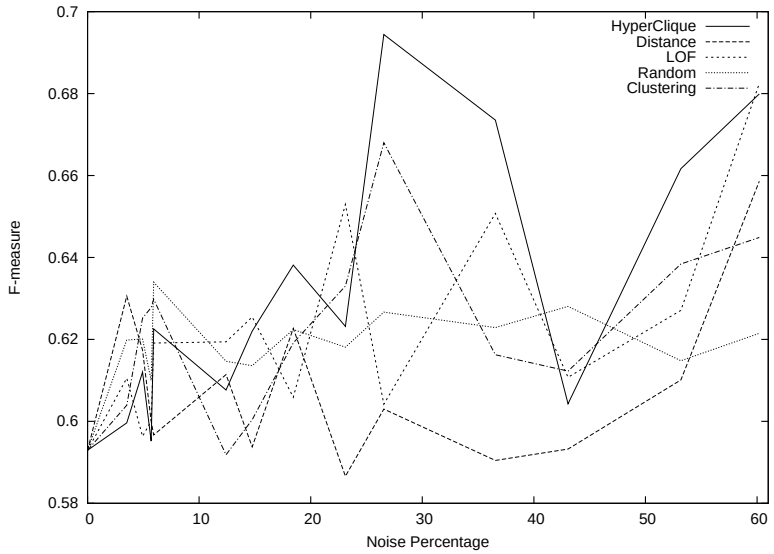


Fig. 7. The impact of noise removal techniques on the performance of clustering analysis for the ADS data set in terms of F-measure.

performance for some continuous data sets when the number of clusters is specified as the actual number of classes in the data or the right MinPts has been identified for LOF. We also found that the distance-based noise removal technique does not perform well for any of the tested data sets. This may imply that the data sets we used have regions of varying density.

C. The Impact of Noise Removal on Association Analysis

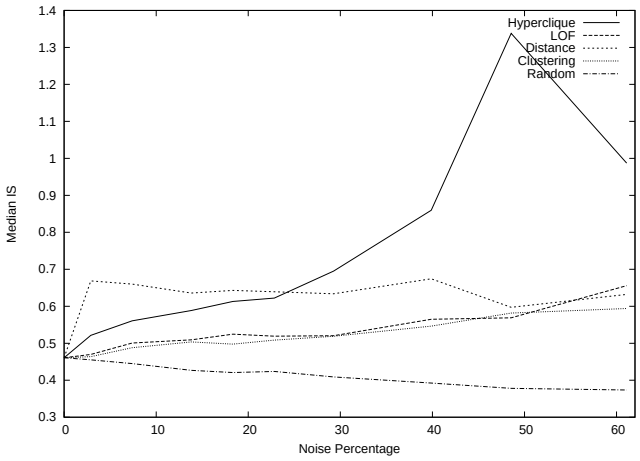
Here, we evaluate the impact of noise removal on the results of association analysis. We used the following experimental procedure for this evaluation. Since each noise cleaning technique can result in

a data set with a different range of supports for the examples, the median of these supports was used as the minimum support for deriving closed frequent patterns. However, when this support failed to give sufficiently many patterns, this threshold had to be reduced by multiplying it by a factor between 0 and 1. The exact value of this factor for each data set is listed in the last column of Table VI. Finally, the quality of the closed frequent patterns over the examples derived at this refined threshold was evaluated using the median of the IS of the patterns.

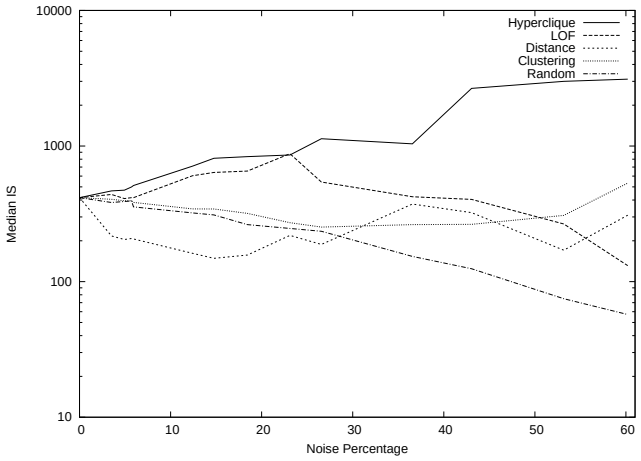
Figures 8 (a) and (b) show the median of the IS of the resulting association patterns for the WEST5 and ADS data sets. For the WEST5 data set, we observe that HCleaner provides the best association results compared to other noise removal techniques when the percentage of noise objects is above 25%. Slightly better results can be observed for the ADS data set. In this case, HCleaner provides the best performance for all ranges of noise percentages considered.

Figures 9 (a) and (b) show the median of the IS of the resulting association patterns for the RE0 and OH8 data sets, respectively. As shown in these two figures, HCleaner can achieve better performance when a large portion of noise has been removed. However, the performance of all these noise removal techniques is relatively close for the OH8 data set when the percentage of noise objects is less than $\sim$50%. The performance of HCleaner is worse than other noise removal techniques for the RE0 data set when the percentage of noise objects is low.

Also, as previously noted, every association rule derived from a hyperclique pattern will have a confidence value above the h-confidence threshold. In other words, if association rules are generated from objects covered by hyperclique patterns identified using a relatively high h-confidence threshold, it is guaranteed that association rules with low confidence will not be generated. In this sense, HCleaner can help generate association rules with better quality, since the specified h-confidence threshold is a lower bound for the confidence of these association rules. In contrast, there are no theoretical guarantees for the quality of association rules for other outlier based noise removal methods.

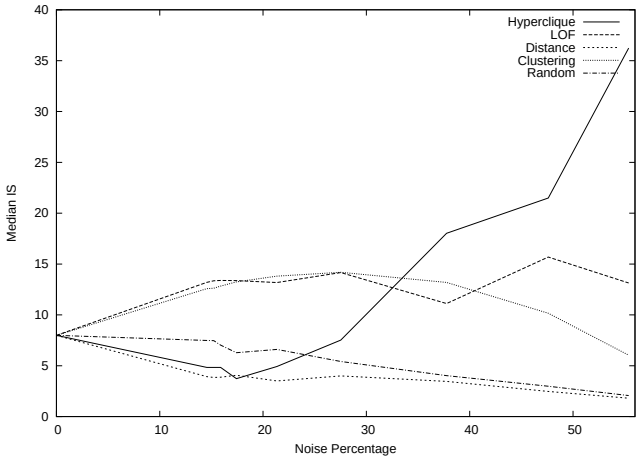


(a) The WEST5 document data set.

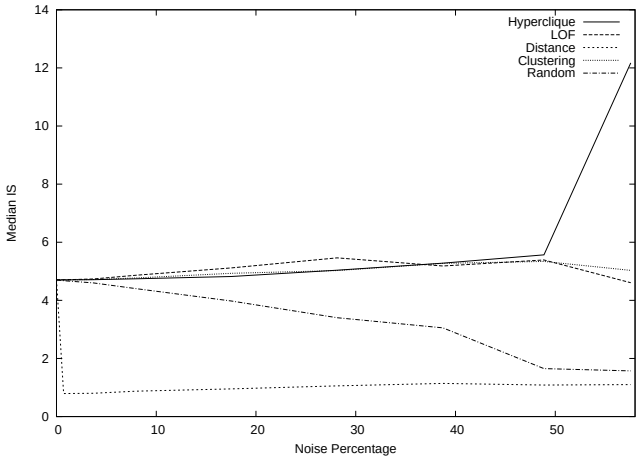


(b) The ADS document data set.

Fig. 8. The impact of noise removal on the results of association analysis for WEST5 and ADS in terms of the IS measure.



(a) The RE0 document data set.



(b) The OH8 document data set.

Fig. 9. The impact of noise removal on the results of association analysis for RE0 and OH8 in terms of the IS measure.

D. Sensitivity Analysis

In this subsection, we discuss several issues related to parameter selection for CCleaner and HCleaner. Figure 10 (a) shows the performance of CCleaner for clustering analysis as the number of clusters is changed. For the OH8 data set, the best noise removal is achieved when the number of clusters equals the actual number of classes (10). Also, it can be observed that the performance of CCleaner declines as the number of clusters gets farther from the actual number of classes in the data. In other words, the performance of CCleaner is very sensitive to the specified number of clusters. If the number of clusters is very small, then this approach has performance similar to that of the distance based approach. If the

number of clusters is very large, then this approach becomes similar to the random approach for removing noise. For this reason, CCleaner tend to perform poorly when the number of clusters deviates from the actual number of classes in the data. Often, the actual number of classes is not known. Thus, if CCleaner is used for noise removal, our results may not be as good as the results presented in Figures 4, 5, and 6, where the number of clusters was set to the actual number of classes.

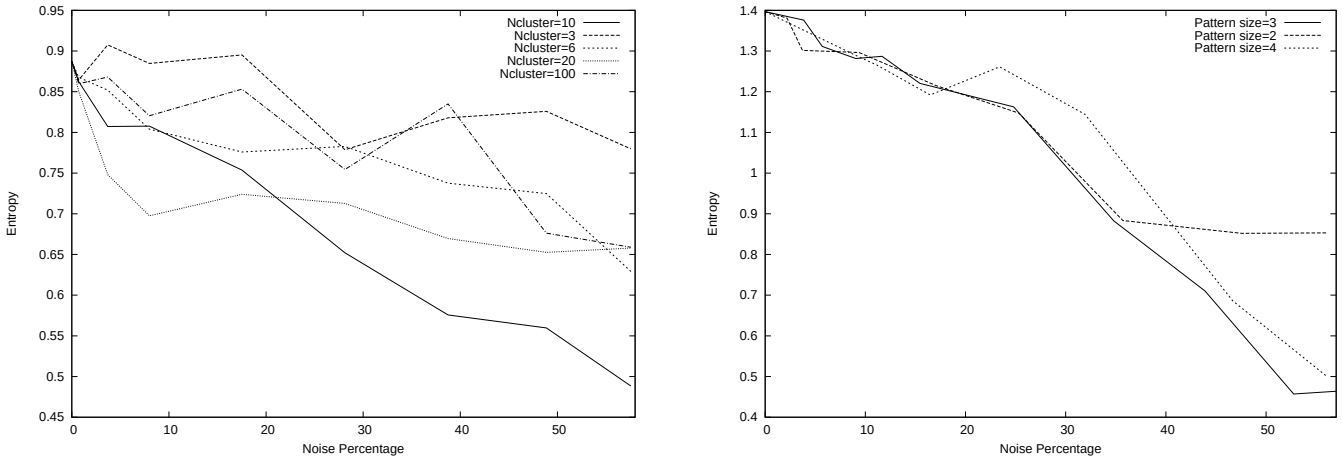
Figure 10 (b) shows the performance of HCleaner for clustering analysis when hyperclique patterns of different sizes are used for filtering out noise objects. In the figure, we observe that the best performance is obtained when size-3 hyperclique patterns are used as filters for HCleaner. Also, we notice that there are only small performance differences among size-2, size-3, and size-4 hyperclique patterns when the percentage of noise objects is low, while size-3 patterns tends to perform better for a large percentage of noise objects. Finally, there is a tradeoff between the size of patterns and the coverage of data objects. If we use hyperclique patterns with a size more than three, HCleaner tends to filter out many more data objects for a specified h-confidence threshold. Thus, to obtain the same percentage of noise objects eliminated, the h-confidence threshold needs to be reduced, and this may result in poorer performance. In contrast, if we use size-2 hyperclique patterns, a very high h-confidence threshold may eliminate only a small number of noise objects. Therefore, more strongly-correlated pairs of objects can be captured and better results can be achieved with size-3 patterns.

While we have only illustrated the parameter sensitivity results of CCleaner and HCleaner for two data sets, similar results hold for the other data sets used in our experiments.

VII. Conclusions

The goal of the work presented in this paper is to boost the quality of data analysis, and capture the underling patterns in the data by reducing the effect of noise at the data analysis stage. This may be noise due to imperfections in the data collection process or noise that consists of irrelevant or weakly relevant data objects. Our focus was on data sets with a very high level of noise.

We provided a framework for evaluating the effectiveness of noise removal techniques for enhancing



(a) The OH8 document data set.

(b) The WEST5 document data set.

Fig. 10. The effect of the number of clusters on the performance of CCleaner for OH8 and WEST5 with respect to entropy.

data analysis that is based on the hypothesis that better noise removal yields better data analysis. Our study was restricted to unsupervised data mining techniques at the data analysis stage.

As part of this work, we studied the performance of four techniques. Three of these techniques were based on existing outlier detection methods. We also proposed a hyperclique based data cleaner (HCleaner). We evaluated these four data cleaning techniques by comparing how much they improved the results of clustering and association analysis. As demonstrated by our experimental results, HCleaner tends to have better noise removal capabilities than the outlier based approaches. Although CCleaner and LOF had good performance in some situations, their performance was not as consistent. In addition, HCleaner has a superior capability for generating higher quality association patterns.

There are several directions for future work. First, given that HCleaner, CCleaner, and the LOF based method were each the best in different situations, it could be useful to consider a voting scheme that combines these three techniques. Also, we would like to investigate the impact of these noise removal techniques on classification performance.

ACKNOWLEDGMENTS

This work was partially supported by NSF grant # IIS-0308264, NSF grant # ACI-0325949, and by Army High Performance Computing Research Center under the auspices of the Department of the Army,

Army Research Laboratory cooperative agreement number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD*, 1993.

[2] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *Proceedings of the Sixth European Conference on the Principles of Data Mining and Knowledge Discovery*, 2002.

[3] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38, New York, NY, USA, 2003. ACM Press.

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. Lof:identifing density based local outliers. In *Proc. of the 200 ACM SIGMOD International Conference on management of Data*, 2000.

[5] Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[6] www.dictionary.com.

[7] Michael B. Eisen, Paul T. Spellman, Patrick O. Browndagger, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 95:25, 1998.

[8] Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of Third SIAM International Conference on Data Mining, San Francisco, CA, USA*, May 2003.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discoverying clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[10] A. Gavin et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147, 2002.

[11] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[12] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. Ajax: An extensible data cleaning tool. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000.

[13] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C. Saita. Declarative data cleaning: Language, model and algorithms. In *Proceedings of the 2001 Very Large Data Bases (VLDB) Conference*, 2001.

[14] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 73–84. ACM Press, June 1998.

[15] Eui-Hong Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, B. Mobasher, and Jerry Moore. Webace: A web agent for document categorization and exploration. In *Proc. of the 2nd International Conference on Autonomous Agents*, 1998.

[16] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 127-138, May 1995.

[17] M.A. Hernandez and S.J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowldge Discovery*, 2:9–37, 1998.

[18] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126, 2004.

[19] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series. Prentice Hall, Englewood Cliffs, New Jersey, March 1988. Book available online at http://www.cse.msu.edu/∼jain/Clustering_Jain_Dubes.pdf.

[20] George Karypis. Cluto: Software for clustering high dimensional datasets. /www.cs.umn.edu/∼karypis.

[21] E. M. Knorr, R. T .Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal: Very Large Databases*, 8:237–253, 2000.

[22] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324, 1997.

[23] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM Press, 1999.

[24] M. L. Lee, T. W. Ling, and W. L. Low. Intelliclean: A knowledge-based intelligent data cleaner. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[25] D. Lewis. Reuters-21578 text categorization text collection 1.0. In *http://www.research.att.com/ lewis*, 1997.

[26] Infoshare Limited. Best value guide to data standardization. *InfoDB*, July 1998, Available from http://www.infoshare.ltd.uk.

[27] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proc. of the ACM-SIGMOD Workshop on Research Issues on Knowledge Discovery and Data Mining*, 1997.

[28] K. Orr. Data quality and systems theory. *CACM*, 41:66–71, 1998.

[29] M. F. Porter. An algorithm for suffix stripping. In *Program, 14(3)*, 1980.

[30] Leonid Portnoy, Eleazar Eskin, and Salvatore J. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, 2001.

[31] S. Ramaswamy, R. Rastogi, and S. Kyuseok. Efficient algorithms for mining outliers from large data sets. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2000.

[32] T. Redman. The impact of poor data quality on the typical enterprise. *CACM*, 41:79–82, 1998.

[33] C. J. Van Rijsbergen. *Information Retrieval (2nd Edition)*. Butterworths, London, 1979.

[34] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[35] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of International Conference on Very Large Databases*, 1998.

[36] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right objective measure for association analysis. *Inf. Syst.*, 29(4):293–313, 2004.

[37] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson Addison-Wesley, 2005.

[38] TREC. Text retrieval conference. In *http://trec.nist.gov*.

[39] Hui Xiong, Pang-Ning Tan, and Vipin Kumar. Mining hyperclique patterns with confidence pruning. In *Technical Report 03-006, Department of computer science, University of Minnesota - Twin Cities*, January 2003.

[40] Hui Xiong, Pang-Ning Tan, and Vipin Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proceedings of the third IEEE International Conference on Data Mining*, pages 387–394, 2003.

[41] Yiming Yang. Noise reduction in a statistical approach to text categorization. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR*, pages 256–263. ACM Press, 1995.

[42] Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors, *KDD*, pages 296–305. ACM, 2003.

[43] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 103–114. ACM Press, 1996.

Hui Xiong is an assistant professor in the Management Science and Information Systems department at Rutgers, the State University of New Jersey. He received the B.E. degree in Automation from the University of Science and Technology of China, China, the M.S. degree in Computer Science from the National University of Singapore, Singapore, and the Ph.D. degree in Computer Science from the University of Minnesota, MN, USA. His research interests include data mining, statistical computing, Geographic Information Systems (GIS), Biomedical informatics, and information security. He has published over 20 technical papers in peer-reviewed journals and conference proceedings and is the co-editor of the book entitled ”Clustering and Information Retrieval”. He has also served on the program committees for a number of conferences and workshops. Dr. Xiong is a member of the IEEE Computer Society and the ACM.

Gaurav Pandey Gaurav Pandey earned his B.Tech. degree in Computer Science and Engineering from the Indian Institute of Technology, Kanpur, India. He is currently a PhD student in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. His research interests include data mining, machine learning and bioinformatics.

Michael Steinbach Michael Steinbach earned his B.S. degree in Mathematics, a M.S. degree in Statistics, and an M.S. degree in Computer Science from the University of Minnesota. He is currently a PhD student in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. He is a co-author of Introduction to Data Mining and has published numerous technical papers in peer-reviewed journals and conference proceedings His research interests include data mining, statistics, and bioinformatics. He is a member of the IEEE Computer Society and the ACM.

Vipin Kumar is currently the Director of Army High Performance Computing Research Center and Professor of Computer Science and Engineering at the University of Minnesota. His research interests include High Performance computing and data mining. He has authored over 200 research articles, and co-edited or coauthored 9 books including the widely used text book “Introduction to Parallel Computing”, and ”Introduction to Data Mining” both published by Addison-Wesley. Kumar has served as chair/co-chair for many conferences/workshops in the area of data mining and parallel computing, including IEEE International Conference on Data Mining (2002) and 15th International Parallel and Distributed Processing Symposium (2001). Kumar serves as the chair of the steering committee of the SIAM International Conference on Data Mining, and is a member of the steering committee of the IEEE International Conference on Data Mining. Kumar serves or has served on on the editorial boards of Knowledge and Information Systems, IEEE Computational Intelligence Bulletin, Annual Review of Intelligent Informatics, Parallel Computing, the Journal of Parallel and Distributed Computing, IEEE Transactions of Data and Knowledge Engineering (93-97), IEEE Concurrency (1997-2000), and IEEE Parallel and Distributed Technology (1995-1997). He is a Fellow of IEEE, a member of SIAM, and ACM, and a Fellow of the Minnesota Supercomputer Institute.